FE[r]Chain: Enforcing Fairness in Blockchain Data Exchanges Through Verifiable Functional Encryption

Camille Nuoskala
camille.nuoskala@tuni.fi
Tampere University
Tampere, Finland

Reyhaneh Rabbaninejad
reyhaneh.rabbaninejad@tuni.fi
Tampere University
Tampere, Finland

Tassos Dimitriou
tassos.dimitriou@ieee.org
Kuwait University
Kuwait City, Kuwait

Antonis Michalas*
antonios.michalas@tuni.fi
Tampere University
Tampere, Finland

ABSTRACT

Functional Encryption (FE) allows users to extract specific function-related information from encrypted data while preserving the privacy of the underlying plaintext. Though significant research has been devoted to developing secure and efficient Multi-Input Functional Encryption schemes supporting diverse functions, there remains a noticeable research gap in the development of verifiable FE schemes. Functionality and performance have received considerable attention, however, the crucial aspect of verifiability in FE has been relatively understudied. Another important aspect that prior research in FE with outsourced decryption has not adequately addressed is the fairness of the data-for-money exchange between a curator and an analyst. This paper focuses on addressing these gaps by proposing a verifiable FE scheme for inner product computation. The scheme not only supports the multi-client setting but also extends its functionality to accommodate multiple users – an essential feature in modern privacy-respecting services. Additionally, it demonstrates how this FE scheme can be effectively utilized to ensure fairness and atomicity in a payment protocol, further enhancing the trustworthiness of data exchanges.

CCS CONCEPTS

• **Security and privacy** → *Key management*; **Access control**; **Privacy-preserving protocols**; Public key encryption.

KEYWORDS

Blockchain, Fairness, Functional Encryption, Verifiable Decryption

1 INTRODUCTION

Functional Encryption (FE) is a powerful cryptographic paradigm that revolutionizes secure data processing by enabling fine-grained access control and computation on encrypted data. This innovative approach aims to strike a delicate balance between privacy preservation and functionality, allowing data owners to selectively disclose specific information while keeping the remainder confidential.

FE schemes leverage a key generation algorithm that yields decryption keys with remarkable capabilities. Each decryption key dk_f is associated with a function f. Unlike traditional cryptographic techniques, applying dk_f to a ciphertext $\mathsf{Enc}(x)$ does *not* recover the original value x, but instead reveals the evaluation $f(x)$, while preserving the confidentiality of x. Initially, FE constructions enabled computation over a *single* ciphertext, but recent advancements [8, 22] have introduced a more versatile notion known as multi-input FE (MIFE). In MIFE schemes, given ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)$, a user can utilize dk_f to obtain the result of $f(x_1, \ldots, x_n)$.

Although significant research efforts in FE have focused on constructing secure and efficient MIFE schemes and supporting various functions, there is a noticeable gap in the literature when it comes to developing verifiable FE schemes. The need for verifiable FE schemes arises from the requirement to ensure the integrity and trustworthiness of the decryption process, particularly in scenarios involving untrusted decryptors or when interacting with potentially compromised entities. Addressing this gap in research is essential as it advances the field of FE toward more comprehensive and trustworthy solutions, bridging the divide between functionality and verifiability. Through verifiable decryption, a user can verify that the functional decryption was honestly computed without having access to the underlying decryption key. This feature is of paramount importance in cases where the decryptor is an untrusted third party. Satisfying the property of verifiable decryption in FE, is an important step towards assuming stronger threat models, by removing trust from, traditionally, fully trusted entities.

Fair Payments: Another important issue that has not been addressed adequately by prior research in the context of FE with outsourced decryption is the fairness of the data-for-money exchange between a curator and an analyst. The curator collects user-encrypted data and presents them in an appropriate form to an analyst, who wishes to purchase a weighted average of these data, under the assumption that neither of them should be trusted. If a malicious analyst is given the functional decrypted results he may cheat the provider and not pay. After all, he owns the decrypted data thus violating fairness. For a similar reason, the analyst cannot pay in advance as the curator may never release the promised data. In this work we leverage the verifiability of functional decryption to develop a protocol that ensures that no party can cheat the other; in particular, data is delivered *if and only if* an appropriate payment is made, essentially making the protocol *atomic*.

While fair exchange poses a fundamental challenge that necessitates the involvement of a trusted third party, our approach utilizes

*Also with RISE Research Institutes of Sweden.

the Blockchain network to eliminate the requirement for such an intermediary. Specifically, we introduce a payment protocol that leverages blockchain technology to establish a secure and transparent framework for transactions between a curator and an analyst. This protocol enables the exchange of functional output for monetary compensation, ensuring a fair and atomic payment process that eliminates the need for trust in the curator or any external intermediaries. Furthermore, through the implementation of simple blockchain transactions where most of the data is transferred off-chain, we not only streamline the payment process but also reduce costs, making it more economical and efficient.

Use cases: We consider *data marketplaces* as a practical motivation for our protocols. The development of services that have access to large amounts of user-generated data has led to the emergence of data-exchange marketplaces, where owners of data can receive rewards for the data they offer to third parties. One example includes crowd-sensing applications [25], in which data collected through devices of participating users are analyzed and made available to requestors or the broader public. Another example includes the smart electricity grid in which "smart meters", devices deployed at home locations, can provide real-time monitoring of electricity consumption thus balancing the use of energy and costs and benefiting both users and electricity providers [18].

Marketplace applications of this form typically use a service provider or curator who administers the data-sharing infrastructure. At a high level, the curator receives user data, processes them accordingly, and delivers aggregates to requestors of this information. However, the success of these data marketplaces depends on addressing several key concerns about data confidentiality, output verifiability, and fairness.

- *Data Condidentiality*: The key factor to the success of these applications is user participation. However, collecting information from user devices or smart meters has important privacy implications, since contributed data may be strongly related to user activities and daily routines. Guaranteeing data privacy and confidentiality is of utmost importance to ensure that sensitive data is not exposed or misused by third parties.
- *Verifiability of results*: As requestors of information rely on the accuracy and quality of the purchased data to make informed decisions, ensuring that the provided data is verifiable is crucial to maintaining trust between the data curator and buyers. Hence no buyer or requestor of data should be tricked to purchase wrong or falsified results.
- *Fairness* is also a key consideration, ensuring that data curators receive appropriate compensation for supplied data and that buyers will get the data they paid for. As the existence of a centralized authority that monitors the exchange of money-for-data is not desirable, it is important that the right tools are developed to ensure that no party can cheat the other.

By prioritizing the verifiability of results, atomicity of transactions, and fairness, our verifiable FE protocol can be used to foster trust, encourage participation, and promote the growth of reliable marketplaces for data exchange.

Contribution: Our contribution can be summarized as follows:

C1: We endow the FE protocol of Castagnos *et al.* [13] with a *multi-client* feature. This new construction aims at allowing the users and the key curator a better management of the ciphertexts accessible by the analyst.

C2: Our construction includes a *verification* method to ensure the correctness of the computation the analyst receives. Additionally, we provide a variant to extend the protocol to n users, encrypting their data separately.

C3: We also propose a blockchain-based payment protocol between a curator and an analyst that can be used to trade the functional output for an appropriate amount. Our payment protocol ensures fairness and atomicity of payments without placing any trust in the curator or other third parties. Additionally, it uses only simple blockchain transactions thus reducing cost and enhancing the efficiency of the payment process.

2 RELATED WORK

(Verifiable) Functional Encryption: FE was first introduced by Boneh *et al.* in [9] as a generalization of Public-Key Encryption (PKE). Since then, numerous new definitions and different approaches have been proposed [22, 23, 30]. As its name suggests, functional encryption schemes are usually implemented with a limited set of functions for a specific use case. In our work, we focus on Inner Product Functional Encryption (IPFE), in which the function is the inner product for a given vector. Prior to our work, numerous IPFE protocols have been proposed, based on different hardness assumptions. Among these, the *learning with error* (LWE) problem has been very popular [1, 2, 6] for its security against quantum attacks. However, the structure of the ciphertexts makes it difficult to design an efficient verification method, and *none* of these cryptosystems has been endowed with such a feature. Another popular approach is the use of the Decisional Diffie-Hellman problem (DDH) [2, 6, 13, 24], or related assumptions. The interest of these constructions is the efficiency and simplicity of their algorithms. Moreover, the ciphertexts being elements of a cyclic group, designing a verification feature based on *zero-knowledge proof* (ZKP) on a discrete logarithm is extremely practical.

Meanwhile, several recent works address the problem of verifiable decryption in FE and propose generic methods. Among them, we cite the work of Badrinarayanan *et al.* [7] that formally defines for the first time the need for a verification method for FE. They provide a method of constructing a verifiable protocol from any FE scheme, including IPFE. However, this construction is primarily theoretical and lacks efficiency in terms of real use-case application. Subsequently, recent protocols have introduced a more pragmatic approach [28, 29], showing that verifiable IPFE can also be practical. However, the use cases they consider primarily focus on malicious users. This differs from our approach.

As this works address the trust issues between two parties, it is important to mention the existence of *decentralized* IPFE [3, 14]. In a decentralized setup, data holders do not rely on a trusted authority for key management, which solves the privacy issues of classical IPFE. This field has been widely studied in the past few years and, despite being more resource-demanding than previous schemes, some practical works have been designed [16]. However, this setup requires constant online participation from the users, as well as being more computationally expensive.

The purpose of this article is to demonstrate the practicability of verifiable FE in a real use case scenario, namely fairness in data marketplaces. We chose to work on the centralized construction of Castagnos *et al.* [13], which has the advantage of proposing a protocol with efficient decryption hence more suitable to our use case. Thereunder, we present some background knowledge and state-of-the-art in contingent payments.

Contingent Payments: Several works in the literature use the blockchain to exchange data and rewards, where a seller can sell any verifiable information for some payment in cryptocurrencies. For example, in [26], a smart contract is used to handle transferred data and payments by the collector. A similar approach is followed in [19], however, the underlying assumption is that data consumers and the service providers do not collude with each other. Fairswap [20] tries to eliminate these problems through proofs of misbehavior that aim at reducing the transaction cost in case of a cheating participant.

A characteristic of smart contract-based solutions is that typically all data have to go through the smart contract, increasing monetary costs and resulting in rather inefficient systems. Our work overcomes this by requiring only the actual payment transaction to use the blockchain network while most of the protocol steps take place *offchain.* We do this by relying on hash time-lock scripts (HTLC), a simpler solution for trustless fair payment, that can be used to transfer assets to the recipientif the pre-image of a target hash value is provided. However, to achieve this, the FE protocol must be enhanced with a *verifiability* property. We consider this another important contribution of this work.

Similarly to our proposal, Zero Knowledge Contingent Payments [27] can be used to exchange a secret for a fee. This was later extended in [11], allowing a seller to receive payment once a certain service has been rendered. Our work resembles these protocols in the sense that the secret to be exchanged is the key that can be used to unlock the purchased FE data.

3 PRELIMINARIES

This section outlines the requirements for understanding this paper. We first define the notation and mathematical background necessary for the FE constructions before introducing the concept of blockchain, which serves as the foundation for our protocol.

Notation: For $m \in \mathbb{N}^*$, $[m]$ is the set $\{1,\dots,m\}$. Vectors are denoted in bold lowercase letters. The inner product between two vectors $\mathbf{x} = (x_1,\dots,x_n)$ and $\mathbf{y} = (y_1,\dots,y_n)$ is $\langle \mathbf{x},\mathbf{y}\rangle = x_1y_1 + \dots + x_ny_n$ and their Hadamard product is $\mathbf{x} \circ \mathbf{y} = (x_1y_1,\dots,x_ny_n)$. For a set $\mathcal{X}$, we use $x \overset{\$}{\leftarrow} \mathcal{X}$ if x is sampled uniformly at random from $\mathcal{X}$ and $x \leftarrow \mathcal{D}_{\mathcal{X},\sigma}$ if x is sampled from the standard Gaussian distribution in $\mathcal{X}$ for deviation σ. A PPT adversary $\mathcal{ADV}$ is a randomized algorithm for which there exists a polynomial $P(x)$ s.t. for all input x, the running time of $\mathcal{ADV}(x)$ is bounded by $|P(x)|$. A function $f : \mathbb{N} \mapsto \mathbb{R}$ is said negligible if $\forall c \in \mathbb{N}, \exists x_0 \in \mathbb{N}$ such that $\forall x \geq x_0, f(x) < x^{-c}$. We denote $negl(\cdot)$ an arbitrary negligible function.

3.1 Inner Product Functional Encryption

An Inner Product Functional Encryption (IPFE) scheme enables users to calculate the inner product between a vector of plaintexts and a vector for which they possess the functional decryption key, without exposing the plaintexts.

Definition 3.1 (Inner Product Functional Encryption (IPFE)). An IPFE scheme for a message space $\mathcal{M}$ and a vector space $\mathcal{Y}$ is a tuple $\mathsf{IPFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ such that:

- $\mathsf{Setup}\left(1^\lambda\right)$: The setup algorithm Setup is a probabilistic algorithm that on input the security parameter λ, outputs a master public and private key pair $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$: The encryption algorithm Enc is a probabilistic algorithm that on input the master public key mpk and a message $\mathbf{x} = (x_1,\dots,x_n) \in \mathcal{M}$, outputs a ciphertext $\mathbf{c}$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$: The key generation algorithm KeyGen is a deterministic algorithm that on input the master secret key msk and a vector $\mathbf{y} \in \mathcal{Y}$, outputs a functional decryption key $\mathsf{dk}_{\mathbf{y}}$.
- $\mathsf{Dec}\left(\mathsf{dk}_{\mathbf{y}}, \mathbf{c}\right)$: The decryption algorithm Dec is a deterministic algorithm that on input a functional decryption key $\mathsf{dk}_{\mathbf{y}}$ and a ciphertext $\mathbf{c}$, outputs the inner product of $\mathbf{y}$ on the plaintexts $\langle \mathbf{x},\mathbf{y}\rangle$.

An IPFE scheme is said to be correct if the following equality holds:

$$Pr[\mathsf{Dec}\left(\mathsf{dk}_{\mathbf{y}}, \mathbf{c}\right) \neq \langle \mathbf{x},\mathbf{y}\rangle \mid [(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}\left(1^\lambda\right)] \wedge$$
$$[\mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{mpk},\mathbf{x})] \wedge [\mathsf{dk}_{\mathbf{y}} \leftarrow \mathsf{KeyGen}(\mathsf{msk},\mathbf{y})]] = negl(\lambda)$$

Remark that the decryption key in definition 3.1 depends only on the vector $\mathbf{y}$ and not on any specific ciphertext. This means that anyone with access to the functional decryption key $\mathsf{dk}_{\mathbf{y}}$ can retrieve the quantity $\langle \mathbf{x},\mathbf{y}\rangle$ for any message $\mathbf{x}$. However, some scenarios require narrowing the capability of a key to a determined set of data. One way to achieve this is to create a dependency between the decryption key and the encryption, which makes $\mathsf{dk}_{\mathbf{y}}$ specific to a restrained set of ciphertexts. An FE scheme with this feature is called *multi-client.*

Definition 3.2 (Multi-Client Inner Product Functional Encryption). An MCIPFE scheme for a message space $\mathcal{M}$ and a vector space $\mathcal{Y}$ is a tuple $\mathsf{MCIPFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ such that:

- $\mathsf{Setup}\left(1^\lambda\right)$: The setup algorithm Setup is a probabilistic algorithm that on input the security parameter λ, outputs a master public and private key pair $(\mathsf{mpk}, \mathsf{msk})$;
- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \ell)$: The encryption algorithm Enc is a probabilistic algorithm that on input the master public key mpk, a message $\mathbf{x} = (x_1,\dots,x_n) \in \mathcal{M}$ and a label ℓ, outputs a ciphertext $\mathbf{c}^{(\ell)}$;
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}, \ell)$: The key generation algorithm KeyGen is a deterministic algorithm that on input the master secret key msk, a vector $\mathbf{y} \in \mathcal{Y}$ and a label ℓ, outputs a functional decryption key $\mathsf{dk}_{\mathbf{y}}$;
- $\mathsf{Dec}\left(\mathsf{dk}_{\mathbf{y}}^{(\ell)}, \mathbf{c}^{(\ell')}\right)$: The decryption algorithm Dec is a deterministic algorithm that on input a functional decryption key $\mathsf{dk}_{\mathbf{y}}^{(\ell)}$ and a ciphertext $\mathbf{c}^{(\ell')}$, outputs the inner product $\langle \mathbf{x},\mathbf{y}\rangle$ if $\ell = \ell'$, and $\perp$ otherwise.

In the remainder of the article, we avoid the notations $\mathsf{dk_y}^{(\ell)}$ and $\mathbf{c}^{(\ell)}$ if there is no ambiguity, and use the classical notation $\mathsf{dk_y}$ and $\mathbf{c}$ instead.

The notion of MCIPFE originally comes from the need of users to encrypt their data separately, hence the name *multi-client.* Different clients could be registered in the same system but encrypt their data under their own encryption key, or label. In this article, we take this feature for another purpose: restraining the set of ciphertext accessible to a given decryption key. The construction that follows this definition is detailed in section 5.2.

3.2 Zero-knowledge proofs

In this paper, we use non-interactive zero-knowledge (NIZK) to prove knowledge and relations over discrete logarithm values without the need for interaction. A zero-knowledge proof is an interactive protocol between a prover P and a verifier V, where the prover tries to convince the verifier about the validity of a statement without the verifier learning anything beyond this fact. Zero-knowledge proofs can be made non-interactive using the Fiat-Shamir heuristic.

We will be using the following standard notation introduced by Camenisch *et al.* [10] in describing such a proof. We will denote by:

$$\pi = NIZK\{(r, u, v) : C = g^r h^u \wedge I = g^v\}$$

a non-interactive zero-knowledge proof, where the prover tries to convince the verifier that it knows r, u and v such that $C = g^r h^u$ and $I = g^v$. The variables r, u, v inside parentheses will denote private values, while C and I on the right will constitute public information available to the verifier V. Once π is created, V can test the validity of the proof submitted by the prover.

3.3 Blockchain

We assume the existence of a public ledger (blockchain) which will be used as the means to exchange data for payments. Payments are made and sent to addresses that are created privately by users. Addresses are bound to user public keys and transactions must be signed to be considered authentic. The ledger keeps track of the list of coins associated with each address and is considered to be a trustworthy bulletin board that only allows for appending new transactions.

The ledger is updated based on user transactions and provides an interface Transfer (A, B, v) that enables the transfer of v coins from one address A (with an associated verification key pk_A) to another address B (with an associated verification key pk_B), as long as the secret key sk_B is known. This interface allows users to transact among themselves.

As the ledger is typically a linked list data structure, any attempt to modify an existing transaction will result in a chain of updates; however this cannot happen without consensus by the majority of the peers maintaining the ledger. This property gives the blockchain its immutable and transparent character.

4 SYSTEM AND THREAT MODEL

System Model: The system model we consider consists of three entities: *(i)* Key Curator (C), *(ii)* Users ($\mathcal{U}$), *(iii)* Analyst ($\mathcal{A}$).

- **Key Curator** (C)**:** We assume the existence of an authority C, which is responsible for the Setup phase and the generation of the decryption keys upon request of an analyst for a specific vector.
- **Users** ($\mathcal{U}$)**:** Let $\mathcal{U} = u_1, \ldots, u_n$ be a set of users holding sensitive data. They each encrypt their data under the public parameters generated by the curator and send them to the latter. The n users involved in the system individually encrypt their sensitive data into ciphertext using the public parameters generated by C. Then, they send their encrypted data to the curator.
- **Analyst** ($\mathcal{A}$)**:** We consider an external data analyst $\mathcal{A}$ who expects the result of selective computation on the data held by the users. $\mathcal{A}$ interacts with the curator and sends functional queries to C .

Threat Model: The curator C generates the master key through the Setup phase and administers the data collection infrastructure. We consider a typical crowd-sensing scenario in which users submit their data and obtain rewards for them (see for example [17]). Hence, C has access to all data. However, consumers of these data are *not* expected to trust the curator to provide them with the correct results. Therefore, we do not require C to be trusted by an analyst $\mathcal{A}$ and consider the possibility that C sends the wrong decryption key to mislead the results of the analyst or that he does not send the data at all during the transaction. On the user side, we reasonably consider that the users are honest in the data they provide, but they are not compelled to trust each other.

In terms of security, we assume that the curator does not collude with the analyst, as this would provide the analyst with access to all user data. This assumption is reasonable in our use case since we address the problem of exchanging data for a *fee* between the curator and the analyst.

Concerning this exchange, we expect the scheme to meet the following security requirements (i) *Atomicity/Fairness*: either both the analyst and the curator respectively get a correct decryption and payment, or the transaction is aborted and neither party gets anything. Essentially, it is crucial to ensure that the curator is not paid without the correct computation of the functional query, and the analyst cannot evade payment after obtaining the correct computation result from the curator. (ii) *Confidentiality*: computation results should be protected against eavesdroppers or malicious entities looking to acquire information about plaintexts associated with ciphertexts.

We highlight again that users trust the curator with their data as they have already been rewarded for them. Furthermore, this is required in a classical IPFE setup: given the capability to generate a decryption key for arbitrary vectors, C can compute trivial inner products on the ciphertexts hence recovering the plain data. The construction of IPFE schemes that do not rely on a trusted authority requires a decentralized setup [3, 14]. In this type of construction, the users themselves generate the decryption keys which implies a constant online involvement and is not suitable for the fair-exchange application we propose.

5 CONSTRUCTING A VERIFIABLE MCIPFE

Consider a scenario where a data analyst wants to perform selective computations on sensitive data $\mathbf{x} = (x_1, \ldots, x_n)$ held by a single user. In this scenario, a key curator runs the Setup algorithm, sets mpk as public information, and keeps msk secret. The user encrypts the message $\mathbf{x}$ into a ciphertext $\mathbf{c} = (c_1, \dots c_n) \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathbf{x})$. The analyst, who has access to $\mathbf{c}$, wants to calculate $\langle \mathbf{y}, \mathbf{x} \rangle$ for a vector $\mathbf{y}$. Therefore, she sends a request to the curator to obtain the functional decryption key $\mathsf{dk}_\mathbf{y}$. The curator, given the analyst's vector $\mathbf{y}$ and secret key msk, produces the decryption key $\mathsf{dk}_\mathbf{y}$ through KeyGen and sends it back to the analyst. Finally, the latter can use Decrypt to obtain the result.

In this scenario, we require the decryption key $\mathsf{dk}_\mathbf{y}$ to be specific to a ciphertext or set of ciphertexts. This restriction permits better access control and is recommended in the use case we consider, where an analyst purchases specific results. This can be achieved using an MCIPFE construction that we defined in definition 3.2.

Further down, we begin with the basic scheme of Castagnos *et al.* [13] and note its limitations and security concerns for our system model. Then we extend it by adding a multi-client feature in section 5.2 and multi-user support in section 5.3. Finally, we consider verifiable decryption in section 5.4.

5.1 Construction of Castagnos *et al.*

To present our contribution, we first recall the IPFE scheme of Castagnos *et al.* [13] in protocol 1. Mark that the system model supported by this protocol admits a single user.

In previous constructions [1, 4, 15] based on Additive-ElGamal encryption scheme [5], the decryption procedure relies on computing a discrete logarithm in a cyclic group. This problem is known to be hard in the general case, hence the correctness of the protocols requires length restrictions on the exponent size. In terms of practical use, this appears to be a serious issue, as it requires strict limitations on the size of the messages as well as the set of decryption keys that can be delivered to analysts. The IPFE construction presented in this section relies on a DDH group with an easy DL subgroup. In short, it consists in generating a cyclic group G together with a subgroup F of G such that the discrete logarithm is easy to compute in F but hard in G. We detail this primitive in appendix A.

First of all, the curator runs the Setup algorithm to generate public parameters $\mathsf{pp} = (p, \tilde{s}, f, g, h)$, secret keys $\mathbf{s} = (s_1, \dots, s_n)$, $\mathbf{t} = (t_1, \dots, t_n)$ and public keys $(h_1, \dots, h_n)$. The user generates a noise r and encrypts data with Encrypt algorithm, i.e. the message x_i is encrypted into $c_i = f^{x_i} h_i^r$ under the public key h_i. The user eventually obtains a ciphertext $\mathbf{c} = (c_0, c_1, \dots, c_n)$, where c_0 is a hint for the decryption.

Protocol 1 (IPFE over $\mathbb{Z}$ under the DDH-f assumption [13]). *Following definition 3.1, this protocol consists in 4 algorithms* (Setup, Encrypt, KeyGen, Decrypt)*, described in fig. 1.*

The pair of algorithms (Gen, Solve) *is detailed in definition A.2 and the choice of the deviation σ is discussed together with the security analysis in appendix B.*

```
Setup $(1^\lambda, 1^\mu, n)$:
Set $\mathsf{pp} := (p, \tilde{s}, g, f, G, F) \leftarrow \mathsf{Gen}(1^\lambda, 1^\mu)$
Sample $\alpha \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathbf{s}, \mathbf{t} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$
for $1 \le i \le n$ do
    Compute $h_i \leftarrow g^{s_i} h^{t_i}$
end
return $\mathsf{msk} = (\mathbf{s}, \mathbf{t})$ and $\mathsf{mpk} = (\mathsf{pp}, h := g^\alpha, h_1, \dots, h_n)$

Encrypt $(\mathsf{mpk}, \mathbf{x})$:
Sample $r \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$
for $1 \le i \le n$ do
    Compute $c_i \leftarrow f^{x_i} h_i^r$
end
return $\mathbf{c} = (c_0 = (g^r, h^r), c_1, \dots c_n)$

KeyGen $(\mathsf{msk}, \mathbf{y})$:
Compute $\mathsf{s}_\mathbf{y} \leftarrow \langle \mathbf{y}, \mathbf{s} \rangle$ and $\mathsf{t}_\mathbf{y} \leftarrow \langle \mathbf{y}, \mathbf{t} \rangle$
return $\mathsf{dk}_\mathbf{y} = (\mathbf{y}, \mathsf{s}_\mathbf{y}, \mathsf{t}_\mathbf{y})$

Decrypt $(\mathsf{mpk}, \mathsf{dk}_\mathbf{y}, \mathbf{c})$:
Compute $\mathsf{c}_\mathbf{y} \leftarrow \prod_{i=1}^n c_i^{y_i} / ((g^r)^{\mathsf{s}_\mathbf{y}} (h^r)^{\mathsf{t}_\mathbf{y}})$
Compute $\mathsf{sol} \leftarrow \mathsf{Solve}(p, \mathsf{pp}, \mathsf{c}_\mathbf{y})$
if $\mathsf{sol} \ge p/2$ then
    $\mathsf{sol} \leftarrow \mathsf{sol} - p$
return $\mathsf{sol}$
```

Figure 1: Castagnos *et al.* IPFE algorithms [13]

Correctness: Let us verify that $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{dk}_\mathbf{y}, \mathbf{c})$ *indeed outputs* $\langle \mathbf{y}, \mathbf{x} \rangle$.

$$\begin{aligned}
\mathsf{c}_\mathbf{y} &= \prod_{i=1}^{n} c_i^{y_i} / \left((g^r)^{\mathsf{s}_\mathbf{y}} (h^r)^{\mathsf{t}_\mathbf{y}} \right) \\
&= \prod_{i=1}^{n} \left(f^{x_i} (g^{s_i} \cdot h^{t_i})^r \right)^{y_i} / \left(g^{r\langle \mathbf{y}, \mathbf{s} \rangle} h^{r\langle \mathbf{y}, \mathbf{t} \rangle} \right) \\
&= f^{\langle \mathbf{y}, \mathbf{x} \rangle} g^{r\langle \mathbf{y}, \mathbf{s} \rangle} h^{r\langle \mathbf{y}, \mathbf{t} \rangle} / \left(g^{r\langle \mathbf{y}, \mathbf{s} \rangle} h^{r\langle \mathbf{y}, \mathbf{t} \rangle} \right) = f^{\langle \mathbf{y}, \mathbf{x} \rangle}
\end{aligned}$$

Then compute $\mathsf{Solve}(p, \mathsf{pp}, \mathsf{c}_\mathbf{y}) = \mathsf{Solve}\left(p, \mathsf{pp}, f^{\langle \mathbf{y}, \mathbf{x} \rangle}\right) = \langle \mathbf{y}, \mathbf{x} \rangle$.

On the other hand, consider an external user, e.g. a data analyst, who wants to perform computations on the data $\mathbf{x}$. Since this protocol is designed for the inner product, the output can be $\langle \mathbf{y}, \mathbf{x} \rangle = y_1 x_1 + \ldots + y_n x_n$ for a valid vector $\mathbf{y}$ chosen by the analyst. The analyst then sends a request $\mathbf{y}$ to the curator, who runs the KeyGen algorithm to provide the functional decryption key $\mathsf{dk}_\mathbf{y} = (\langle \mathbf{y}, \mathbf{s} \rangle, \langle \mathbf{y}, \mathbf{t} \rangle)$, where $\mathbf{s}, \mathbf{t}$ are the secret keys. By running the Decrypt algorithm, the analyst can finally retrieve the desired quantity $\langle \mathbf{y}, \mathbf{x} \rangle = y_1 x_1 + \ldots + y_n x_n$.

Limitations: We motivate the construction of protocol 2 by highlighting several limitations of the previous construction concerning the properties expected by our system model. First of all, the protocol of Castagnos *et al.* is not multi-client, i.e. the functional decryption key $(\langle \mathbf{y}, \mathbf{s} \rangle, \langle \mathbf{y}, \mathbf{t} \rangle)$ generated by the curator is not specific to a ciphertext, hence giving the possibility to the data analyst to get more information than the one she originally requested. Secondly, this scheme has been designed for a set of linearly independent requested vectors small enough not to endanger the security of the plaintexts, as detailed below. The set of a functional decryption key is much bigger in a MCIPFE scheme, due to label dependency. However, allowing the generation of numerous decryption keys brings forth a different security vulnerability on the secret keys themselves. Finally, this protocol only supports a single user. Hence we design in protocol 3 a variant supporting n users encrypting the data independently.

Multi-Client Feature: The multi-client construction presented in definition 3.2 allows a user to encrypt messages through different labels and divide the set of encrypted data into smaller subsets. In protocol 2, this feature allows the curator to add a dependency between the decryption key and the ciphertext. Therefore, an analyst holding a decryption key $\mathsf{dk_y}^{(\ell)} \leftarrow \mathsf{KeyGen}(msk, \mathbf{y}, \ell)$ can only compute $\langle \mathbf{y}, \mathbf{x} \rangle$ for $\mathbf{x}$ encrypted under the same label ℓ. Consecutively, to restrict the access of a data analyst to a set of data, it is necessary to change the label of further encrypted data. In particular, if the curator wishes to sell a single computation, it is possible to deliver a functional decryption key specific to a single ciphertext encrypted with a specific label.

5.2 Adding Multi-Client Support

Now, we introduce our corresponding MCIPFE construction following definition 3.2. We propose a new decryption key generation allowing the curator to output a functional decryption key that depends on a specific set of ciphertexts, encrypted under a common label. First of all, this construction requires a preliminary encoding of this label.

Label Encoding: To construct an MCIPFE, we design an encryption and decryption procedure that allows a user to encrypt a ciphertext under a specific label $\ell \in \mathbb{N}$. To ensure this specificity hence the security of the protocol, it is necessary to verify that, given a decryption key for a label ℓ, an adversary can not forge a decryption key for another label ℓ'. As the decryption key from protocol 1 is an inner product, the multi-client decryption key can not be linear in ℓ. Therefore, we consider an encoding method $\mathsf{Ecd} : \mathbb{Z} \rightarrow \mathbb{Z}^n; \ell \mapsto (\ell_1, \ldots, \ell_n)$ where each ℓ_i depends non-linearly on ℓ. This idea has already been proposed by Abdalla *et al.* [2] for the general case using on a *pseudo-random function* $\mathsf{PRF}(i, \ell)$ as a basis for their encoding. However, this general approach is not the best choice for our protocol which relies on a public encoding. Therefore, we believe that the use of a hash function would be sufficient to guarantee the non-linearity of the encoding and the unforgeability of a label ℓ, as detailed in theorem B.4, while providing better performances and better flexibility on the choice of labels.

Specification: we use a simple first and second pre-image resistant cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_{n \cdot p}$. The output of the hash h can be split afterwards into n components of range p with a canonical projection $\pi : h := h_{n-1} \| \ldots \| h_0 \mapsto (h_{n-1}, \ldots, h_0)$.

Eventually, given as input a label ℓ, we define the following encoding algorithm:

$$\begin{aligned} \mathsf{Ecd} : \{0,1\}^* &\rightarrow \mathbb{Z}_{n \cdot p} \rightarrow \mathbb{Z}_p^n \\ \ell &\mapsto H(\ell) \mapsto (\ell_1, \ldots, \ell_n) \end{aligned}$$

Protocol 2 (MCIPFE scheme under the DDH-f assumption). *Our multi-client variant of protocol 1, consists of 4 algorithms* (Setup, Encrypt, KeyGen, Decrypt) *described in fig. 2.*

```
Setup $\left(1^\lambda, 1^\mu, n\right)$:
Set $\mathsf{pp} = (p, \tilde{s}, g, f, G, F) \leftarrow \mathsf{Gen}\left(1^\lambda, 1^\mu\right)$
Pick $\alpha \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathbf{s}, \mathbf{t} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$
for $1 \le i \le n$ do
    Compute $h_i \leftarrow g^{s_i} h^{t_i}$
end
return $\mathsf{msk} = (\mathbf{s}, \mathbf{t})$ and $\mathsf{mpk} = (\mathsf{pp}, h := g^\alpha, h_1, \ldots, h_n)$

Encrypt $(\mathsf{mpk}, \mathbf{x}, \ell)$:
Pick $r \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$
Encode $(\ell_1, \ldots, \ell_n) \leftarrow \mathsf{Ecd}(\ell)$
for $1 \le i \le n$ do
    Compute $c_i \leftarrow f^{x_i} h_i^{r \ell_i}$
end
return $\mathbf{c}^{(\ell)} = (c_0 := (g^r, h^r), c_1, \ldots, c_n)$

KeyGen $(\mathsf{msk}, \mathbf{y}, \ell)$:
Encode $(\ell_1, \ldots, \ell_n) \leftarrow \mathsf{Ecd}(\ell)$
Compute $\mathsf{s_y} \leftarrow \sum_{i=1}^n s_i y_i \ell_i$ and $\mathsf{t_y} \leftarrow \sum_{i=1}^n t_i y_i \ell_i$
return $\mathsf{dk_y}^{(\ell)} = (\mathbf{y}, \mathsf{s_y}, \mathsf{t_y})$

Decrypt $\left(\mathsf{mpk}, \mathsf{dk_y}^{(\ell')}, \mathbf{c}^{(\ell')}\right)$:
Compute $\mathsf{c_y} \leftarrow \prod_{i=1}^n c_i^{y_i} / \left((g^r)^{\mathsf{s_y}} (h^r)^{\mathsf{t_y}}\right)$
Compute $\mathsf{sol} \leftarrow \mathsf{Solve}(p, \mathsf{pp}, \mathsf{c_y})$
if $\mathsf{sol} \ge p/2$ then
    $\mathsf{sol} \leftarrow \mathsf{sol} - p$
return $\mathsf{sol}$
```

Figure 2: MCIPFE algorithms

The pair of algorithms (Gen, Solve) *is detailed in definition A.2 and the choice of the deviation σ is discussed together with the security analysis in appendix B.*

Correctness: Let us verify that $\mathsf{Decrypt}\left(\mathsf{mpk}, \mathsf{dk_y}^{(\ell')}, \mathbf{c}^{(\ell)}\right)$ *indeed outputs* $\langle \mathbf{y}, \mathbf{x}\rangle$ *if and only if* $\ell = \ell'$.

$$\begin{aligned}
\mathsf{c_y} &= \prod_{i=1}^{n} c_i^{y_i} / \left((g^r)^{\mathsf{s_y}} (h^r)^{\mathsf{t_y}} \right) \\
&= \prod_{i=1}^{n} \left(f^{x_i} \left(g^{s_i} \cdot h^{t_i} \right)^{r\ell_i} \right)^{y_i} / \left(g^{r \sum s_i y_i \ell_i'} h^{r \sum t_i y_i \ell_i'} \right) \\
&= f^{\langle \mathbf{y},\mathbf{x} \rangle} g^{r \sum s_i y_i \ell_i} h^{r \sum t_i y_i \ell_i} / \left(g^{r \sum s_i y_i \ell_i'} h^{r \sum t_i y_i \ell_i'} \right) \\
&= f^{\langle \mathbf{y},\mathbf{x} \rangle} g^{r \sum s_i y_i (\ell_i - \ell_i')} h^{r \sum t_i y_i (\ell_i - \ell_i')}
\end{aligned}$$

Then compute $\mathsf{sol} \leftarrow \mathsf{Solve}\left(p, \mathsf{pp}, \mathsf{c_y}\right)$ *that is*

$$\begin{aligned}
\mathsf{sol} &= \mathsf{Solve}\left(p, \mathsf{pp}, f^{\langle \mathbf{y},\mathbf{x} \rangle} g^{r \sum s_i y_i (\ell_i - \ell_i')} h^{r \sum t_i y_i (\ell_i - \ell_i')}\right) \\
&= \langle \mathbf{y}, \mathbf{x} \rangle \ \ \textit{iff} \ \ g^{r \sum y_i (\ell_i - \ell_i')(s_i + \alpha t_i)} = 1
\end{aligned}$$

This last equality is obviously achieved if $\ell = \ell'$. *In the case* $\ell \neq \ell'$, *some values of* $(r, \mathbf{y}, \mathbf{s}, \alpha, \mathbf{t})$ *can inevitably output a valid result. However, we argue in appendix B that without the knowledge of* $r, \alpha, \mathbf{s}$ *and* $\mathbf{t}$, *a malicious analyst can neither forge such a tuple* $(r, \mathbf{y}, \mathbf{s}, \alpha, \mathbf{t})$ *nor verify the validity of an output, rendering the decryption useless. Considering this observation, we assume that the algorithm outputs* $\perp$ *if* $\ell \neq \ell'$. *This concludes the correctness.*

This construction being similar to the previous one, the curator starts by running the Setup algorithm to generate the public parameters $(p, \tilde{s}, f, g, h)$, the secret keys $\mathbf{s}, \mathbf{t}$ and the public keys $h_1, \ldots, h_n$. As mentioned earlier, the user chooses a label ℓ and encrypts data with Encrypt, generating a ciphertext $\mathbf{c} = (c_0, c_1, \ldots, c_n)$, where c_0 is a hint used for the decryption. Now, consider an analyst wishing to retrieve $\langle \mathbf{y}, \mathbf{x} \rangle$ for a vector $\mathbf{y}$. She sends the curator a request $\mathbf{y}$ for the ciphertext $\mathbf{c}$ of $\mathbf{x}$, which provides the functional decryption key $\mathsf{dk_y} = (\sum s_i y_i \ell_i, \sum t_i y_i \ell_i)$ – output of KeyGen, for the label ℓ under which $\mathbf{x}$ is encrypted. By running the Decrypt algorithm, the analyst can finally retrieve the desired quantity $\langle \mathbf{y}, \mathbf{x} \rangle = y_1 x_1 + \cdots + y_n x_n$.

5.3 Supporting Multiple Users

The proposed protocol features the multi-input paradigm, allowing a user to evaluate a multi-linear function, namely the inner product, on a vector of plaintexts $\mathbf{x} = (x_1, \ldots, x_n)$. To fully exploit this construction, we extend the algorithms to n different users, who encrypt their data separately. In short, for users $u_1, \ldots, u_n$, we generate a master public key with n components $\mathsf{mpk} = (\mathsf{pk_1}, \ldots, \mathsf{pk_n})$ so that a user u_i can produce a ciphertext c_i encrypting the plaintext x_i under the key $\mathsf{pk_i}$. These ciphertexts are then merged into a vector $\mathbf{c} = (c_1, \ldots, c_n)$ for an analyst to perform computations on plaintexts produced by different users.

Protocol 3 (MCIPFE scheme for multiple users). *The* n*-users variant of our MCIPFE protocol 2 consists in a set of users* $u_1, \ldots, u_n$ *and four algorithms* (Setup, Encrypt, KeyGen, Decrypt) *described in fig. 3.*

```
Setup $\left(1^\lambda, 1^\mu, n\right)$:
Set $\mathsf{pp} = (p, \tilde{s}, g, f, G, F) \leftarrow \mathsf{Gen}\left(1^\lambda, 1^\mu\right)$
Pick $\alpha \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathbf{s}, \mathbf{t} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$
for $1 \le i \le n$ do
    Compute $h_i \leftarrow g^{s_i} h^{t_i}$
end
return $\mathsf{msk} = (\mathbf{s}, \mathbf{t})$ and $\mathsf{mpk} = (\mathsf{pp}, h := g^\alpha, h_1, \ldots, h_n)$
Encrypt $((u_1, \ldots, u_n), \mathsf{mpk}, \mathbf{x}, \ell)$:
Encode $(\ell_1, \ldots, \ell_n) \leftarrow \mathsf{Ecd}(\ell)$
for each user $u_i$, $i \in [n]$ do
    Pick $r_i \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and compute $c_i \leftarrow f^{x_i} h_i^{r_i \ell_i}$
    $u_i$ outputs $(g^{r_i}, h^{r_i}, c_i)$
end
Set $c_0 := ((g^{r_1}, \ldots, g^{r_n}), (h^{r_1}, \ldots, h^{r_n}))$
return $\mathbf{c} := (c_0, c_1, \ldots, c_n)$
KeyGen $(\mathsf{msk}, \mathbf{y}, c_0)$:
Encode $(\ell_1, \ldots, \ell_n) \leftarrow \mathsf{Ecd}(\ell)$
Compute $\mathsf{dk_y} \leftarrow \prod_{i=1}^{n} (g^{r_i})^{s_i y_i \ell_i} (h^{r_i})^{t_i y_i \ell_i}$
return $\mathsf{dk_y}$
Decrypt $\left(\mathsf{mpk}, \mathsf{dk_y}^{(\ell')}, \mathbf{c}^{(\ell)}\right)$:
Compute $\mathsf{c_y} \leftarrow \prod_{i=1}^{n} c_i^{y_i} / \mathsf{dk_y}^{(\ell')}$
Compute $\mathsf{sol} \leftarrow \mathsf{Solve}(p, \mathsf{pp}, \mathsf{c_y})$
if $\mathsf{sol} \ge p/2$ then
    $\mathsf{sol} \leftarrow \mathsf{sol} - p$
return $\mathsf{sol}$
```

Figure 3: Multi-users MCIPFE algorithms

Correctness: Let us verify that $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{dk}_y, \mathbf{c})$ *indeed outputs* $\langle \mathbf{y}, \mathbf{x} \rangle$.

$$\begin{aligned}
\mathsf{c_y} &= \prod_{i=1}^{n} c_i^{y_i} / \left(\prod_{i=1}^{n} (g^{r_i})^{s_i y_i \ell_i'} (h^{r_i})^{t_i y_i \ell_i'} \right) \\
&= \prod_{i=1}^{n} \left(f^{x_i} \left(g^{s_i \ell_i} \cdot h^{t_i \ell_i} \right)^{r_i} \right)^{y_i} / \left(g^{\sum r_i s_i y_i \ell_i'} h^{\sum r_i t_i y_i \ell_i'} \right) \\
&= f^{\langle \mathbf{y},\mathbf{x} \rangle} g^{\sum r_i s_i y_i \ell_i} h^{\sum r_i t_i y_i \ell_i} / \left(g^{\sum r_i s_i y_i \ell_i'} h^{\sum r_i t_i y_i \ell_i'} \right) \\
&= f^{\langle \mathbf{y},\mathbf{x} \rangle} \ \ \textit{if} \ \ell_i = \ell_i' \ , \ \forall i \in [n]
\end{aligned}$$

Finally compute $\mathsf{Solve}\left(p, \mathsf{pp}, \mathsf{c_y}\right) = \mathsf{Solve}\left(p, \mathsf{pp}, f^{\langle \mathbf{y},\mathbf{x} \rangle}\right) = \langle \mathbf{y}, \mathbf{x} \rangle$ *if* $\ell = \ell'$. *We refer the reader to protocol 2 for the case* $\ell \neq \ell'$.

We motivate this construction by the need to encrypt the data *independently*. Indeed, it would be straightforward to construct a multi-user scheme from protocol 2 by having the users to compute a common noise r using *multi-party computation* (MPC) techniques. However, it would require an additional online computation from the users, that does not fit our application case.

5.4 Verifiable decryption for MCIPFE

We now focus on the verification steps we have added to protocol 2. We propose a method that an analyst can use to verify the correctness of the decryption key $\mathsf{dk_y}$ provided by the curator. In case the analyst receives an incorrect value for $\mathsf{dk_y}$ during a transaction, he can either cancel the transaction or request a refund from the curator.

Functional Decryption Key Verification: A cryptographic protocol needs to meet several security assumptions to guarantee data privacy. Among these requirements, achieving ciphertext indistinguishability guarantees that a ciphertext does not reveal any information about the plaintext. However, in such cases, how can an external user confirm that a given ciphertext is a valid encryption? This question is particularly important in the field of operations on encrypted data, such as functional encryption or homomorphic encryption, where an external party requests the output of a specific function. This question has been widely studied [21] due to the growing trend of multi-party computation (MPC).

In fig. 4, we propose a verification method for protocol 2 based on the discrete logarithm (DLOG). This algorithm is followed by a ZKP on DLOG to guarantee no additional leakage is produced on the messages or the secret key. Given an encryption $\mathbf{c}$ of the user's data, assume that an analyst wishes to purchase the encryption $\prod_{i=1}^{n} c_i^{y_i \ell_i}$. Recall that this encryption satisfies the equality

$$\prod_{i=1}^{n} c_i^{y_i \ell_i} = f^{\langle \mathbf{y},\mathbf{x}\rangle} g^{r\sum s_i y_i \ell_i} h^{r\sum t_i y_i \ell_i} = f^{\langle \mathbf{y},\mathbf{x}\rangle} \cdot \mathsf{g_y}$$

To verify that the analyst retrieves the correct result, it is sufficient to verify that the decryption key $\mathsf{dk_y}$ is indeed the right decryption key for the requested vector $\mathbf{y}$. We thereunder design the fig. 4, which allows an analyst to check this correctness exclusively using the information known to her.

```
Verify $(\mathsf{mpk}, \mathsf{dk_y}, c_0 = (g^r, h^r), \ell)$:
Encode $(\ell_1, \ldots, \ell_n) \leftarrow \mathsf{Ecd}(\ell)$
Compute $v \leftarrow \prod_{i=1}^{n} h_i^{y_i \ell_i}$
Compute $\mathsf{g_y} \leftarrow \prod (g^r)^{\mathsf{s_y}} (h^r)^{\mathsf{t_y}}$
return $\mathbb{1}\left[\log_g(v) = \log_{g^r}(\mathsf{g_y})\right]$
```

Figure 4: MCIPFE verification algorithm

To perform the verification, the analyst needs access to the hint c_0 of the ciphertext of interest, with both the public and the functional decryption key. These are all public, hence the verification does not impact the security of the protocol. Considering her vector $\mathbf{y}$, the analyst computes:

$$\prod_{i=1}^{n} h_i^{y_i \ell_i} = \prod_{i=1}^{n} \left(g^{s_i} h^{t_i}\right)^{y_i \ell_i} = \prod_{i=1}^{n} \left(g^{s_i + \alpha t_i}\right)^{y_i \ell_i} = g^{\sum_{i=1}^{n} y_i \ell_i (s_i + \alpha t_i)}$$

On the other hand, mark that:

$$\mathsf{g_y} = g^{r\mathsf{s_y}} h^{r\mathsf{t_y}} = g^{r\sum_{i=1}^{n} s_i y_i \ell_i} h^{r\sum_{i=1}^{n} t_i y_i \ell_i} = g^{r\sum_{i=1}^{n} y_i \ell_i (s_i + \alpha t_i)}$$

The analyst accepts the functional decryption key if $\log_{g^r}(\mathsf{g_y}) = \log_g(v)$ for $v = \prod_{i=1}^{n} h_i^{y_i \ell_i}$.

ZKP on DLOG: Since the discrete logarithm is not easy in G, the analyst cannot compute $\log_{g^r}(\mathsf{g_y})$ or $\log_g(v)$. However, by using a ZKP she can ensure that the two quantities are indeed equal without leaking information about their actual values.

Decryption key verification for multi-users: Considering the multi-users protocol 3, we would like to adapt the verification method to guarantee the analyst the correctness of the decryption key. However, the verification process for multi-users is not as straightforward as the single-user construction, and Figure 4 cannot be extended naturally to the multi-user construction. We therefore propose to substitute the method in Figure 4 with a system of n ZKPs. This is not prohibitive as n ciphertexts c_i have to be transmitted anyway, but does constitute a better method for multi-user key verification. Recall that, for $\mathbf{s} = (s_1, \ldots, s_n)$, $\mathbf{t} = (t_1, \ldots, t_n)$ hidden to the analyst, g^{r_i} for $i \in [n]$ included in c_0, and $\mathbf{y} = (y_1, \ldots, y_n)$ generated by the analyst, the decryption key is of the form $\mathsf{dk_y} = g^{\langle \mathbf{r}\circ\mathbf{y},\mathbf{s}\rangle} h^{\langle \mathbf{r}\circ\mathbf{y},\mathbf{t}\rangle}$, where $\langle \mathbf{r}\circ\mathbf{y}, \mathbf{s}\rangle = \sum r_i y_i s_i$ and $\langle \mathbf{r}\circ\mathbf{y}, \mathbf{t}\rangle = \sum r_i y_i t_i$. Following, we claim that it is sufficient for the analyst to know that the correct secret keys $\mathbf{s}, \mathbf{t}$ were indeed used for the generation of $\mathsf{dk_y}$. Hence we rely on the following ZKP:

$$\pi_i = NIZK\{(s_i, t_i) : h_i = g^{s_i} h^{t_i} \wedge \mathsf{dk_y} = g^{\langle \mathbf{r}\circ\mathbf{y},\mathbf{s}\rangle} h^{\langle \mathbf{r}\circ\mathbf{y},\mathbf{t}\rangle}\}, i \in [n].$$

6 FE[R]CHAIN

Having defined the verifiability aspects of our protocol, we now consider the fairness issue arising by malicious behavior. For example, if the analyst is given $\mathsf{dk_y}$, she may cheat the curator and not pay. After all, she has the decrypted data. Similarly, the analyst may not be motivated to pay in advance as the curator may never release $\mathsf{dk_y}$.

To ensure fairness, C will *not* release $\mathsf{dk_y}$ to the analyst $\mathcal{A}$ but instead will pick a random value k and encrypt $\mathsf{dk_y}$ into $c = (\mathsf{dk_y})^k$. Then it will commit to k and extend the verifiability proof π of the previous section by showing that i) it knows a commitment to k which encrypts $\mathsf{dk_y}$, and ii) $\mathsf{dk_y}$ is correctly formed. In particular, the curator will create a proof π which is given by:

$$\begin{aligned}\pi = NIZK\{(s_i, t_i, k, r, \mathsf{dk_y}) : h_i = g^{s_i} h^{t_i}, i \in [n] \wedge \\ \mathsf{dk_y} = g^{\langle \mathbf{r}\circ\mathbf{y},\mathbf{s}\rangle} h^{\langle \mathbf{r}\circ\mathbf{y},\mathbf{t}\rangle} \wedge \\ c = (\mathsf{dk_y})^k \wedge z = k^d\}\end{aligned}$$

This zero-knowledge proof can be built using conventional sigma protocols to demonstrate the knowledge of representing discrete logarithms. These protocols are founded in the DH assumption and do not rely on any trusted setup [10].

The values c and z as well as the proof π are sent to the analyst *offchain* (notice that $\mathsf{dk_y}$ is still not released). Additionally, the blockchain is not utilized yet. The blockchain will only be used to exchange d for an appropriate amount of money as shown next. In this context, the blockchain operates solely as a medium to facilitate the monetization of d. This approach enables the deferral of costly blockchain operations and optimization of transactions used.

Table 1: Gas cost related to different functions in the smart contract (AVG_GAS_PRICE=32.44 Gwei, 1 ETH = 1854.94 USD)

	GAS	USD
`initiateSwap`	546612	32.9
`completeSwap`	156176	9.4
`refundSwap`	94560	5.69

Getting paid for the data: At this point the analyst $\mathcal{A}$ checks the proof π. If everything verifies, she posts to the blockchain a time-locked transaction $T_{\mathcal{A}\to C}$ which says that $\mathcal{A}$ offers val amount of cryptocurrency to curator C under the condition that "C presents a pre-image d to the committed value z ($= k^d$) within some time window t"; if the conditions are not satisfied the coins return to $\mathcal{A}$. More precisely, the transaction $T_{\mathcal{A}\to C}$ has an output of val coins that can be redeemed by a (future) transaction T if one of the following is true:

(1) T is signed by C and contains a valid d, or
(2) T is signed by $\mathcal{A}$ and the time window t has expired.

The transaction $T_{\mathcal{A}\to C}$ is satisfied if C posts a transaction $T_{C\to A}$ that contains d. This would satisfy condition 1 of $T_{\mathcal{A}\to C}$ and so val coins are transferred to C. If C does not act within the time window t, then $\mathcal{A}$ can sign and post a transaction T that returns the val coins back to her.

This concludes the *onchain* phase of the protocol. Notice that only a single value d is transmitted by the curator! As the transaction does not depend on the actual data to be transferred (since these were exchanged offchain), this makes the protocol very efficient.

Once the analyst receives d, she computes $z^{1/d}$ to recover k first, and $c^{1/k}$ to recover $\mathsf{dk}_{\mathbf{y}}$. Then she uses $\mathsf{dk}_{\mathbf{y}}$ to obtain the final result $f^{\langle \mathbf{x},\mathbf{y}\rangle}$ as explained in the previous sections.

Gas Cost Analysis: We implemented a prototype for the proposed smart contract with three main functions: `initiateSwap`, `completeSwap`, and `refundSwap`. The `initiateSwap` function is utilized by the initiator (analyst $\mathcal{A}$) to initiate a new swap. To create the swap, $\mathcal{A}$ is required to provide various parameters, including a unique swapId, the curator C's address, a commitment to the encryption key, and a timelock value. Additionally, the initiator must transfer the specified amount of money along with the function call.

The `completeSwap` function is invoked by C to disclose the committed key and claim the funds. It verifies that the opened key matches the committed key stored in the contract. Upon successful verification, the funds are transferred to C accordingly. The `refundSwap` function is called by $\mathcal{A}$ to refund the funds if the timelock has expired and the swap has not been completed.

To get gas cost estimations, we tested the contract on Ethereum Ropsten test network. The gas cost results for the deployment and function calls on the smart contract together with their corresponding price in USD are presented in Table 1. These results are calculated based on the ethereum exchange rate 1 ETH = 1854.94 USD and the average gas price 32.44 Gwei = $32.44 \cdot 10^{-9}$ ETH on June 7th, 2023[1].

[1]https://ycharts.com/indicators/ethereum_average_gas_price

Notice that the above protocol can be further optimized to support *multiple* purchases. If the analyst is interested in obtaining many functional results, all of these can be "encrypted" with the same key by the curator. Thus with a single onchain transaction, the analyst can obtain all these different results, effectively reducing the amortized cost per transaction even more.

7 CONCLUSION

In this paper, we designed a multi-client verifiable FE scheme for inner products. While the emphasis in recent research has primarily been on functionality and performance, the importance of verifiability in FE is understudied. Satisfying the property of verifiable decryption in FE, is an important step towards assuming stronger threat models, by removing trust from, traditionally, fully trusted entities. Additionally, our construction's easy decryption procedure makes the overall scheme efficient.

This work further contributes by proposing a design of a blockchain-based payment protocol between a curator and an analyst that can be used to trade the functional output for an appropriate amount. Our payment protocol ensures fairness and atomicity of payments without placing any trust in the curator or third parties. This contribution promotes the practical use of FE schemes.

ACKNOWLEDGMENTS

This work was funded by the HARPOCRATES EU research projects (No. 101069535) and SWARMCHESTRATE (No. 101135012).

REFERENCES

[1] Michel Abdalla, Dario , Dario Fiore, Romain Gay, and Bogdan Ursu. 2018. Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions Without Pairings. In *Advances in Cryptology – CRYPTO 2018.*

[2] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. 2019. From Single-Input to Multi-client Inner-Product Functional Encryption. In *Advances in Cryptology – ASIACRYPT 2019*, Steven D. Galbraith and Shiho Moriai (Eds.). Springer International Publishing, Cham.

[3] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. 2019. Decentralizing Inner-Product Functional Encryption. In *Public-Key Cryptography – PKC 2019*, Dongdai Lin and Kazue Sako (Eds.). Springer International Publishing, Cham, 128–157.

[4] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. 2015. Simple Functional Encryption Schemes for Inner Products. In *Public-Key Cryptography – PKC 2015*, Jonathan Katz (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 733–751.

[5] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. 2015. Simple Functional Encryption Schemes for Inner Products. In *Public-Key Cryptography – PKC 2015*, Jonathan Katz (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 733–751.

[6] Shweta Agrawal, Benoît Libert, and Damien Stehlé. 2016. Fully Secure Functional Encryption for Inner Products, from Standard Assumptions. In *Advances in Cryptology – CRYPTO 2016*, Matthew Robshaw and Jonathan Katz (Eds.). Springer Berlin Heidelberg, 333–362.

[7] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. 2016. Verifiable Functional Encryption. In *Advances in Cryptology – ASIACRYPT 2016*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 557–587.

[8] Alexandros Bakas, Antonis Michalas, and Tassos Dimitriou. 2022. Private Lives Matter: A Differential Private Functional Encryption Scheme. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy* (Baltimore, MD, USA) *(CODASPY '22)*. Association for Computing Machinery, New York, NY, USA, 300–311.

[9] Dan Boneh, Amit Sahai, and Brent Waters. 2011. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*. Springer, 253–273.

[10] Jan Camenisch and Markus Stadler. 1997. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich* 260 (1997).

[11] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. 2017. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM Conference on Computer and Communications*

Security.

[12] Guilhem Castagnos and Fabien Laguillaumie. 2015. Linearly homomorphic encryption from DDH. In *Cryptographers' Track at the RSA Conference*. Springer, 487–505.

[13] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. 2018. *Practical Fully Secure Unrestricted Inner Product Functional Encryption Modulo p.* 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, Proceedings, Part II, 733–764.

[14] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. 2018. Decentralized Multi-Client Functional Encryption for Inner Product. In *Advances in Cryptology – ASIACRYPT 2018*, Thomas Peyrin and Steven Galbraith (Eds.). Springer International Publishing, Cham, 703–732.

[15] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. 2018. Decentralized Multi-Client Functional Encryption for Inner Product. In *Advances in Cryptology – ASIACRYPT 2018*, Thomas Peyrin and Steven Galbraith (Eds.). Springer International Publishing, Cham, 703–732.

[16] Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. 2020. Dynamic Decentralized Functional Encryption. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing.

[17] Tassos Dimitriou and Antonis Michalas. 2022. Incentivizing Participation in Crowd-Sensing Applications Through Fair and Private Bitcoin Rewards. *IEEE Access* 10 (2022).

[18] Tassos Dimitriou and Ameer Mohammed. 2020. Fair and privacy-respecting bitcoin payments for smart grid data. *IEEE Internet of Things Journal* 7, 10 (2020), 10401–10417.

[19] Huayi Duan, Yifeng Zheng, Yuefeng Du, Anxin Zhou, Cong Wang, and Man Ho Au. 2019. Aggregating crowd wisdom via blockchain: A private, correct, and robust realization. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*. IEEE, 1–10.

[20] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. 2018. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 967–984.

[21] Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter Rønne, and Tjerand Silde. 2022. Verifiable Decryption in the Head. In *Information Security and Privacy*, Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo (Eds.). Springer International Publishing, Cham, 355–374.

[22] Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. 2014. Multi-input functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*.

[23] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. 2013. How to run turing machines on encrypted data. In *Annual Cryptology Conference*. Springer, 536–553.

[24] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. 2010. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *Advances in Cryptology – EUROCRYPT 2010*, Henri Gilbert (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–91.

[25] Jinwei Liu, Haiying Shen, Husnu S Narman, Wingyan Chung, and Zongfang Lin. 2018. A survey of mobile crowdsensing techniques: A critical component for the internet of things. *ACM Transactions on Cyber-Physical Systems* 2, 3 (2018), 1–26.

[26] Yuan Lu, Qiang Tang, and Guiling Wang. 2018. Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 853–865.

[27] Gregory Maxwell. 2011. Zero knowledge contingent payment. *Bitcoin Wiki* (2011).

[28] Dinh Duy Nguyen, Duong Hieu Phan, and David Pointcheval. 2023. Verifiable Multi-Client Functional Encryption for Inner Product. Cryptology ePrint Archive, Paper 2023/268.

[29] Najmeh Soroush, Vincenzo Iovino, Alfredo Rial, Peter B. Roenne, and Peter Y. A. Ryan. 2020. Verifiable Inner Product Encryption Scheme. In *Public-Key Cryptography – PKC 2020*, Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas (Eds.). Springer International Publishing, Cham, 65–94.

[30] Brent Waters. 2015. A punctured programming approach to adaptively secure functional encryption. In *Annual Cryptology Conference*. Springer, 678–697.

A DECISIONAL DIFFIE-HELLMAN WITH AN EASY SUBGROUP

This section presents the structure of a DDH-f group and the hard problem our construction relies on. First, recall the classical (Decisional) Diffie-Hellman hardness assumption (DDH).

Definition A.1 (Decisional Diffie-Hellman (DDH)). Let $G = \langle g \rangle$ be a cyclic group or order q, generated by g. Let $h \in G$, and $x_1, x_2 \in \mathbb{Z}_q$. Distinguish between $(g^{x_1}, g^{x_2}, g^{x_1 \cdot x_2})$ and (g^{x_1}, g^{x_2}, h) with non negligible probability.

This problem has been widely studied and used in the field of functional encryption. However, it causes limitations for practical use, in particular for primitive relying on the Additional ElGamal. To overcome these limitations, Castagnos *et al.* [13] has introduced a variant of this problem, relying on a subgroup with easy discrete logarithm.

Definition A.2 (DDH group with an easy DL subgroup). A DDH group with an easy DL subgroup is a pair of algorithms (Gen, Solve) defined as:

- Gen $\left(1^\lambda, 1^\mu\right)$: Takes as input two security parameters λ and μ, for $\lambda \leq \mu$. Outputs public parameters pp $= (p, \tilde{s}, g, f, G, F)$, where p is a random μ-bit prime, $G = \langle g \rangle$ is a finite group of order $n = ps$ and $gcd(p, s) = 1$. Additionally, $\widetilde{s}$ is an upper bound for s such that the distribution of $\{g^r, r \xleftarrow{\$} \{0, \ldots, \tilde{s} \cdot p\}\}$ is computationally indistinguishable from the uniform distribution on G. Moreover, the DL problem is easy on $F = \langle f \rangle$, the subgroup of G of order p.
- Solve (p, pp, X): A deterministic algorithm that solves the DL problem in F in a polynomial time, that is:

$$\Pr[x = x' \parallel \text{pp} \xleftarrow{\$} \text{Gen}(1^\lambda, 1^\mu), x \xleftarrow{\$} \mathbb{Z}_p,$$
$$X \leftarrow f^x, x' \leftarrow \text{Solve}(p, \text{pp}, X)] = 1$$

The cryptographic primitives based on cyclic groups often rely on the DDH hardness assumption. Our scheme inherits a similar problem based on the DDH-f construction.

Definition A.3 (Decisional Diffie-Hellman with an easy DL subgroup (DDH-f)). Let $(p, \tilde{s}, g, f, G, F) \leftarrow \text{Gen}\left(1^\lambda, 1^\mu\right)$. Let $x, y \xleftarrow{\$} \mathbb{Z}_n$ and $u \xleftarrow{\$} \mathbb{Z}_n$. Given g^x and g^y, distinguish g^{xy} from $g^{xy} f^u$ with non negligible probability.

Castagnos and Laguillaumie [12] have studied the DDH-f problem in depth and prove that this assumption, despite being weaker than the classical DDH, is still a hard problem in the general case.

B SECURITY ANALYSIS

This section provides a security analysis for our MCIPFE protocol 2 and extends it to the multi-users variant. Secondly, we prove the security of our use-case protocol regarding the threat model defined in section 4.

B.1 Security of the MCIPFE construction

We start with a reminder of the security definition that an IPFE scheme should meet to give a security analysis for both our construction and the verification.

Definition B.1 (IPFE Indistinguishability-Based Security). For an IPFE scheme IPFE = (Setup, Enc, KeyGen, Dec), a challenger **C** and a PPT algorithm **ADV**, define the IND-CPA game extended to IPFE as follows:

IND-CPA Security

Initialize:

C runs $\mathsf{mpk}, \mathsf{msk} \leftarrow \mathtt{Setup}\left(1^\lambda, 1^\mu\right)$ and sends mpk to **ADV**;

Query:

ADV sends a polynomial number of encryption requests for chosen plaintexts and eventually sends two distinct plaintexts $\mathbf{x}_0, \mathbf{x}_1$ to **C**;

Challenge:

C picks $\beta \xleftarrow{\$} \{0,1\}$ and sends $\mathbf{c}_\beta \leftarrow \mathsf{Enc}\left(\mathsf{mpk}, \mathbf{x}_\beta\right)$ to **ADV**;

Functional query:

ADV sends a polynomial number of encryption requests for freshly chosen plaintexts and a polynomial number of decryption key requests for vectors $\mathbf{y}_1, \ldots, \mathbf{y}_t$ such that $\langle \mathbf{x}_0, \mathbf{y}_i \rangle = \langle \mathbf{x}_1, \mathbf{y}_i \rangle$;

Guess:

ADV outputs a guess $\beta' \in \{0,1\}$ on the value of β with an advantage $\epsilon(\lambda)$.

IPFE is said to be λ-IND-CPA secure if and only if the advantage $\epsilon(\lambda) = \left|\frac{1}{2} - Pr[\beta' = \beta]\right|$ is negligible.

We admit that the original construction of Castagnos *et al.* [13] is IND-CPA secure (theorem B.2).

Theorem B.2. *The IPFE protocol 1 is IND-CPA secure under the DDH-f assumption of definition A.3 for $\sigma > \tilde{s} \cdot p^{3/2} \cdot \lambda^{1/2}$ with $\lceil \log p \rceil = \mu$.*

We now argue that our protocol 2 is IND-CPA secure in the sense of IPFE and prove the following theorem B.3.

Theorem B.3. *Let* MCIPFE *be the construction presented in protocol 2. Then* MCIPFE *is IND-CPA secure under the DDH-f assumption. It follows that its multi-user variant given in protocol 3 is also IND-CPA secure.*

Proof. To prove theorem B.3 we rely on a combination of games (definition B.1), $\mathcal{G}_0$ denoting the IPFE game for protocol 1 and $\mathcal{G}_1$ the IPFE game for protocol 3. We assume two algorithms **A** and **B** are executed simultaneously but independently in which **A** is the adversary of a challenger **C** in game $\mathcal{G}_0$ and **B** is also the challenger of adversary **A** in game $\mathcal{G}_1$. We prove that if **A** wins then **B** also wins, i.e. the advantage of **A** is bounded by the advantage of **B**. Since protocol 1 is assumed to be IND-CPA, the advantage of $\mathcal{B}$ in $\mathcal{G}_0$ is negligible, and so is the advantage of **A** in game $\mathcal{G}_1$, which concludes.

Setup of game $\mathcal{G}_0$: First, **C** generates $\mathsf{mpk}, \mathsf{msk} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^\mu\right)$ and sends mpk to **B**. The latter picks two messages $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{Z}^n$ and sends them to **C**. **C** flips a random coin $b \in \{0,1\}$, encrypts $\mathbf{c_b} \leftarrow \mathsf{Encrypt}\left(\mathsf{mpk}, \mathbf{x_b}\right)$ and sends $\mathbf{c_b}$ back to **B**.

Setup of game $\mathcal{G}_1$: **B** samples at random a label $\ell \xleftarrow{\$} \mathbb{Z}$ and encodes it into $(\ell_1, \ldots, \ell_n) \leftarrow \mathsf{Ecd}\left(\ell\right)$. She creates a new master public key $\mathsf{mpk}^\mathbf{B} \leftarrow (h_1^{\ell_1^{-1}}, \ldots, h_n^{\ell_n^{-1}})$, where ℓ_i^{-1} is the inverse of ℓ_i modulo $p-1$, and sends it to **A**. Remark that $\mathsf{mpk}^\mathbf{B}$ is a valid public key, as $h_i^{\ell_i^{-1}} = g^{s_i \ell_i^{-1}} h^{t_i \ell_i^{-1}}$ and $s_i \ell_i^{-1}$ follows a Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma'}$ where $\sigma' \geq \sigma$. The same stands for t_i. Eventually, she sends $\mathsf{mpk}^\mathbf{B}$ to **A**.

Queries: The challenge ciphertext $\mathbf{c_b}$ is a valid encryption for game $\mathcal{G}_1$ under label ℓ. Therefore, **A** can send functional key queries for vectors $\mathbf{y_1}, \ldots, \mathbf{y_n}$ such that $\langle \mathbf{y_i}, \mathbf{x_0} \rangle = \langle \mathbf{y_i}, \mathbf{x_1} \rangle$. Each corresponding decryption key $\mathsf{sk}_{\mathsf{y_i}} = (\mathbf{y_i}, \mathsf{s}_{\mathsf{y_i}}, \mathsf{t}_{\mathsf{y_i}})$ in game $\mathcal{G}_0$ is a valid decryption key in game $\mathcal{G}_1$ for the label ℓ.

Guess: The game $\mathcal{G}_1$ corresponding to the security game for protocol 2, **A** guesses the value of b with an advantage $\epsilon_\mathbf{A}$. Subsequently, **B** guesses the value of b in game $\mathcal{G}_0$ with the same advantage $\epsilon_\mathbf{B} = \epsilon_\mathbf{A}$. Since we assumed protocol 1 to be IND-CPA secure (theorem B.2), our construction protocol 2 is IND-CPA secure. □

We now address the problem of the unforgeability of the decryption key mentioned in the proof of correctness of protocol 2.

Theorem B.4. *Given a decryption key $\mathsf{dk}_\mathsf{y}^{(\ell')}$ for a vector $\mathbf{y}$ and label ℓ', we show that an analyst $\mathcal{A}$ can not recover a usable result from a ciphertext $\mathsf{c}^{(\ell)}$ for $\ell \neq \ell'$*

Proof. As stated, the decryption algorithm leads to $\langle \mathbf{x}, \mathbf{y} \rangle$ *iff* $g^{r \sum y_i(\ell_i - \ell_i')(s_i + \alpha t_i)} = 1$. Recall that $\mathcal{A}$ knows g^r and ℓ, that are public information from $\mathbf{c}^{(\ell)}$, the public key $g^{s_i + \alpha_i}$ for $i \in [n]$ and (y_i, ℓ_i') that are information she owns. For each $i \in [n]$, $(g^r, g^{s_i + \alpha t_i}, g^{r(s_i + \alpha t_i)})$ is a DDH triplet. According to the DDH hardness assumption, the analyst can not distinguish $g^{r(s_i + \alpha t_i)}$ from g^d for $d \xleftarrow{\$} \mathbb{Z}_q$ with non-negligible probability. Consequently, $\mathcal{A}$ can not adaptively choose $\mathbf{y}$ and ℓ so that $g^{r \sum y_i(\ell_i - \ell_i')(s_i + \alpha t_i)} = 1$, unless $y_i(\ell_i - \ell_i') = 0 \mod p$ for every tuple y_i, ℓ_i, ℓ_i'. However, the hash function used for the encoding being pre-image resistant, it is not possible for $\mathcal{A}$ to choose the label ℓ accordingly. This concludes the proof of unforgeability. □

B.2 Security of the fair payment protocol

Theorem B.5. *Let* SIG = (KeyGen, Sign, Vrfy) *be an unforgeable signature scheme used to sign blockchain transactions, and DLOG is hard. If the proof π is sound and zero-knowledge, then the protocol described in section 6 is a fair payment protocol.*

Proof. Consider first the case of a malicious curator whose goal is to cheat an honest analyst by getting a reward and releasing no data, or by releasing erroneous data. Such an attack can be modeled by a security game (details omitted due to space restrictions) in which the adversary wins the game if *(i)* he can extract a valid key-pair which can now be used to sign a new blockchain transaction, thus redeeming another entity's money, or *(ii)* obtain a payment for the wrong data. The first attack can only happen if forging signatures is possible. Similarly, obtaining payment for invalid data is also infeasible as this would break the verifiability property of the FE scheme or the soundness property of the ZK proof system. Hence in both cases, the malicious curator fails.

In a similar manner, a malicious analyst can cheat if she can recover a valid signing key and use it to sign a bitcoin transaction on behalf of another user. However, this is prevented by the unforgeability of the signature scheme. Neither can she post a transaction containing a smaller payment as the curator would abort and not post the transaction that releases the decryption exponent d. Finally, the security of the DLOG problem and the zero-knowledge property of the proof π guarantee that a malicious analyst cannot recover d from the received commitment or the proof π. $\square$